CS559 Midterm Exam

October 28, 2009, 7:15pm-9:15 pm

This exam is closed book and closed notes.

You will have the entire period (until 9:15) to complete the exam, however it was designed to take less time.

Please write your name and CS login on every page! (we may unstaple the exams for grading) Please do this first (and check to make sure you have all of the pages)

Write numerical answers in fractional form or use radicals (square root symbols) – we would prefer to see $\frac{\sqrt{3}}{2}$ than .866. You should not need a calculator for this exam.

Unless otherwise noted, assume that everything is a right-handed coordinate system and that angles are measured counter clockwise. E.g. to find the direction of rotation, point your thumb along the axis and curl your fingers.

If you need extra space, use the back of a page, but clearly mark what everything is. We may look at your work to determine partial credit. Not providing an answer is not the same as giving a wrong answer. For "forced choice" questions (multiple choice, true false, etc.), wrong answers will be penalized more than no answers.

If you feel a question is ambiguous or requires more information for you to make a good answer, you can state any additional assumptions you needed to arrive at your answer. However, you should probably avoid this: usually it is more a sign of a lack of thought about the question or understanding of the material.

One Free Point: 1pt
Question 1: ___4 pts / -1pt
Question 2: ___ 10pts / -5pts
Question 3: ___ 8pts
Question 4: ___ 9pts / -3pt
Question 5: ___ 12pts / -6pt
Question 6: ___ 10pts
Question 7: ___ 15pts
Question 8: ___ 10pts
Question 9: ___ 6pts / -2pt
Question 10: ____ 15 pts

Mean: 75 High: 98
Median: 77 Low:42

At least 1 person got each question right

Total : 100 pts.

Question 1 (4/-1pts)

What is the effect of transforming a 3D object by the following 4x4 matrix, assuming we are working in homogeneous coordinates?

$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$	a) Translates the object b) Scales the object up (makes it bigger) c) Scales the object down (makes it smaller) d) Produces a perspective view of the object e) Does nothing to the object

Since we divide by W, this matrix is equivalent to one with $\frac{1}{2}$, $\frac{1}{2}$, $\frac{1}{2}$, 1 along the diagonal (i.e: scale by $\frac{1}{2}$). But you probably wouldn't want to do this in practice (since it messes up other things)

Question 2 (10pts, 2/-1 per part)

Can each of the following be achieved with a purely local lighting model?

(for each part write "L" on the line if it can be done with local lighting, or "N" if it is a non-local effect, or leave things blank if you don't want to answer)

A. _N_ Shadows
B. _L_ Shiny metal looks different than shiny plastic
C. _L_ Some objects are shinier than others
D. _N_ We can see the reflection of a bright red wall
E. _L_ We can see the reflection of a bright red light

"Local" means that it only considers 1 point of geometry. So it can consider multiple (and colored lights), complicated surface properties (like materials), but not interactions between objects (like shadows or seeing one object reflected in another).

Question 3 (8pts)

What does it mean for a kernel to be seperable? Why is it a useful property for filter kernels?

A seperable kernel can be factored into kernels of lower dimension that can be applied sequentially to achieve the same effect. So a 2D kernel can be factored into 2 1D kernels (that can each be applied in sequence). (5pts = idea of multiple 1D kernels)

Seperability is an important property since it allows for much more efficient implementation and easier analysis. It is much faster to do 2 1D operations than a 2D operation. (3pts=why good)

Question 4 (9pts, 3/-1 per part)

A student recently asked me what filters photoshop actually uses for upsizing. We can figure this out by using a carefully constructed image as described below.

The test image is created by filling the entire image with (100,100,100), and placing a single pixel of value (200,200,200) at the center. (recall that these pixels will look gray on the screen because R=G=B).

We then triple the size of the image. We assume that the upsizing algorithm works correctly, and that the proper cutoff frequency is used. We simply do not know what type of filter is used.

If you care about the details, the image has to start out an odd size (11x11) so it has a center, and since the edge pixels map to edge pixels, the resulting image is actually 31x31. This means that the center pixel in the input maps exactly to the center pixel in the output.

Here are the different kernels we'll consider:

1) Nearest-neighbor interpolation
2) Tent-filter
3) An interpolating cubic (like Catmull-Rom)
4) An approximating cubic that is not interpolating (like B-Spline or Mitchell-Netravali)

For each of the following, list all of the kernels for which the statement is definitely true. If the answer may or may not be true (for example, if it depends on information that you have not been given) do not list that kernel.

Your answer should be a list of 1-4 numbers, or say "none" if no kernel would make the sentence true.

A. (example) The resulting image will not be the correct size
   none
B. (example) The resulting image will be the right size
   1,2,3,4

A. There are no values in the result with value less than 100,100,100.
   1,2 (3 and 4 have negative lobes, so they may ring and have dark patches)
   Note: if you said 4 is always positive (it might be some of the time), it's OK
   1pt for having 1&2, 1pt for not having 3
B. There will be at least 1 pixel with value exactly 200,200,200.
   1,2,3 (are all interpolating – 4 is not interpolating) (tent filter is usually interpolating, but some people might not have thought that way)
   1pt for 1&3 being there, 1pt for 4 not being listed
C. All pixels will have R=G=B (that is, be gray).
   1,2,3,4 (resampling shouldn't cause color shifts – it applies to each channel)

Question 5 (12 pts, +2/-1 per question)

A graphics system implements an OpenGL-like set of transformation commands like we discussed in class, including the same matrix stacks (with 4x4 transforms) that OpenGL uses. However, the divide-by-w used to convert from homogeneous to regular coordinates uses a faulty division circuit that gives wrong answers when w is not equal to 1. Which of the following things will still work? (for each, write "Yes" if it works, "No" if it doesn't, or leave things blank if you don't want to answer).

A. _Yes_ 3D Translations
B. _Yes_ 3D Rotations
C. _Yes_ Gouraud Shading
D. _No_ 3D Drawing using Perspective Projections but no Z-Buffer
E. _Yes_ 3D Drawing using Orthographic Projections with Z-buffering
F. _Yes_ 3D Drawing using Orthographic Projections without Z-buffering

The only place where we need to have w!=1 is for doing perspective projection. Everything else works OK. Also, since we do lighting calculations before projection, shading should work.

Question 6 (10 pts)

Consider a 3D transformation M (a 4x4 homogeneous coordinate matrix) that is created by composing a rotation and a uniform scale. M maps the unit X vector to (0,1,1), and the unit Y vector to (0,1,-1).

Part A: Where does M map the unit Z vector to?

-sqrt(2), 0, 0 – (note that the two other vectors lie in the X=0 plane, so this vector has to lie along the X axis. Figuring out the direction (+/- 1) can be done with the right hand rule. The magnitude can be determined in the next part.)

4 pts for the X axis, 1 pt for magnitude, 1pt direction (6pts total)

Part B: What is the amount of the scale used to make M?

Sqrt(2) – (note that the two vectors that we gave have this length, so that this vector has this length as well)

4pts (and you got 1 pt above for the magnitude, so it was worth half)

Hint: You don't want to figure out what the angles of the rotation are. In fact, you might not need to figure out what the transformation is.

Question 7: (15 pts)

For the following questions, consider a simple graphics toolkit that works like OpenGL (that is, it has a matrix stack, and the transformation commands post-multiply themselves onto it):

Translate(x,y) – post-multiplies a translation matrix onto the top of the matrix stack

Rotate(a) – rotates (counter clockwise around the origin) by a degrees

Scale(x,y) – scales by x and y.

ReflectY() – reflects around the Y axis (note: this changes the X positions)

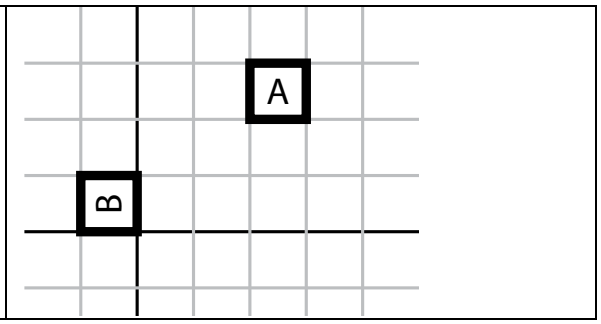
Push() – pushes a copy of the top element on the matrix stack

Pop() – removes the top element from the matrix stack

Block(letter) – draws an "alphabet block" with the letter inside. The block is a unit square, with its lower left corner at (0,0) and its upper right corner at (1,1). You can assume that the letter A is symmetric about its vertical center, and the letters B, C, D and E are symmetric about their horizontal centers.

Here's an example program and its output:

```
Push
Translate(2,2)
Block('A')
Pop
Rotate(90)
Block('B')
```



Here are three functions that draw "arms" given the angles for the "elbow" and "wrist"

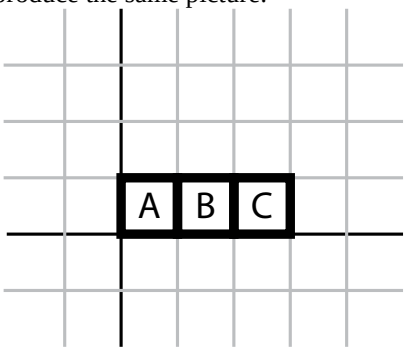
```
arm1(float elbow, float wrist)

      Block('A')
      Translate(1,0)
      Rotate(elbow)
      Block('B')
      Translate(1,0)
      Rotate(wrist)
      Block('C')
```

```
arm2(float elbow, float wrist)
      Block('A')
      Translate(1,1)
      Rotate(elbow)
      Translate(0,-1)
      Block('B')
      Translate(1,0)
      Rotate(wrist)
      Block('C')
```

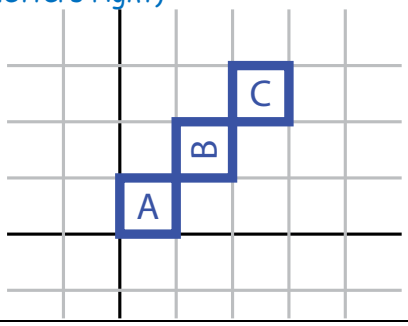
```
arm3(float elbow, float wrist)
      Block('A')
      Translate(1,0)
      Rotate(elbow)
      Block('B')
      Translate(-1,0)
      Rotate(wrist)
      Translate(2,0)
      Block('C')
```

If elbow and wrist are both zero, all three functions produce the same picture:

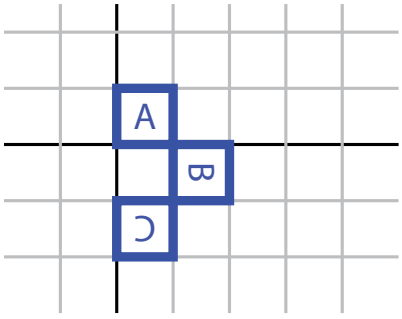


Part B:
What values of wrist and elbow give this picture (that has had the letters removed) when used with arm2? Please write the letters on each block in the picture.

Elbow 90, wrist=-90 (2pt per number, 1 pt letters right)

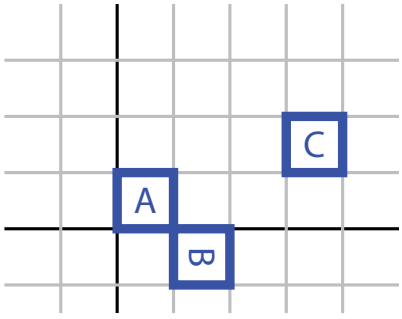


Part A:
Sketch the result of arm1(-90,-90). Be sure to put the letters on the blocks.



1 pt for each block position, 1 pt for B&C orientation

Part C:
Sketch the result of arm3(-90, 90). Be sure to put the letters on the blocks.



1 pt for each block position, 1 pt for B&C orientation

Question 8 (10 pts)

Convolve the discrete signal: f(t) = 0 0 0 0 1 0 0 0 2 2 2 0 0 0 0
(defined over the range 0..14, so that f(0) is the first zero given)

With the discrete kernel: g(t) = -1 1 2 (defined over the range -1..1, so that g(0)=1)

You can assume that both f and g are zero outside of the given ranges

Don't worry about how many leading/trailing zeros you put on the result – just make sure to give all of the non-zero values.

If you reverse the kernel (as it should be), the answer will be:

-1 1 2 0 -2 0 4 6 4

If you don't reverse the kernel (which is actually wrong):

2 1 -1 0 4 6 4 0 -2 (-3pts for correct answer w/non-reversed kernel)

Question 9 (6 pts, 3 per part, -1 for incorrect guess)

Assume that the monitor is carefully gamma corrected.

Note: if you got the wrong answer but gave reasoning for how you got it, we didn't give negative points

9A: Does the difference in *apparent* brightness between a pixel with R=G=B=50 and one with R=G=B=51 appear (to a "normal" viewer) to be larger, smaller, or the same as the difference between 200 and 201?

The same

The point of gamma correction is that each of these steps should look the same to the viewer

9B: If you measured the actual brightness difference between pixels with R=G=B=50 and 51(that is the amount of light energy coming from the monitor) would it be larger, smaller, or the same as the difference between pixels with R=G=B=200 and 201?

(i.e. is I(51)-I(50) (>,<,=) I(201)-I(200) – where I is the measured intensity)

Less

Because the steps are designed to look the same, what must be the same is the relative jump. Since 200 is brighter than 50, then the step must be bigger (in terms of the amount of energy). So 50 is less.

If you got part A wrong, you get 1 point for this being the opposite (e.g. saying it's the same)

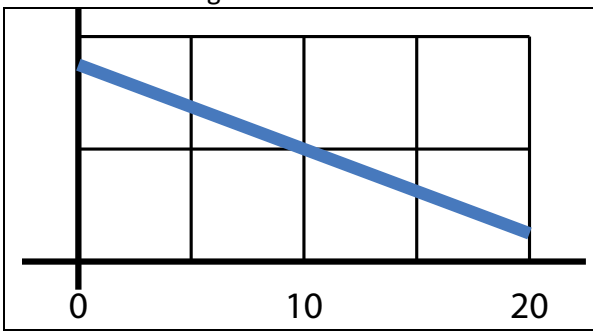
Question 10 (15 pts, 3 per part)

Consider the following scene. A point light source L is placed at (0,10,0). The eye point is placed at (20,10,0). For this question, we're only considering locations with Z=0. Assume we're using the lighting model we discussed in class (what OpenGL does).

Consider the line on the floor from (0,0,0) to (20,0,0), and the brightness on that line caused by the light source. For each of the following lighting conditions, make a graph of the brightness for each X position from 0-20. Note: the actual values of the brightness don't matter (in fact, you can't know exactly what they are since we haven't told you how bright things are, or the exact value of the specular exponent). However, you should be able to create a graph that is qualitatively correct (has the right shape, has the minima and maxima at the right places, …).

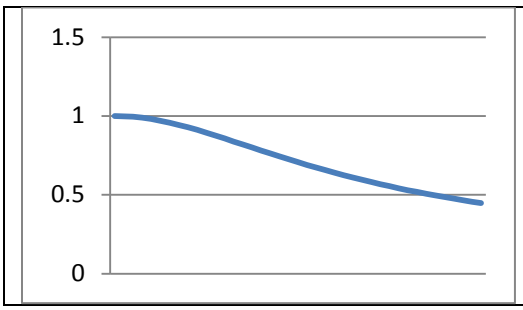
The floor is a *single large triangle* that has vertices at (0,0,0) and (20,0,0).

A) There is no ambient or specular lighting (their values are 0), but a large *diffuse* component, and *Gouraud* shading.



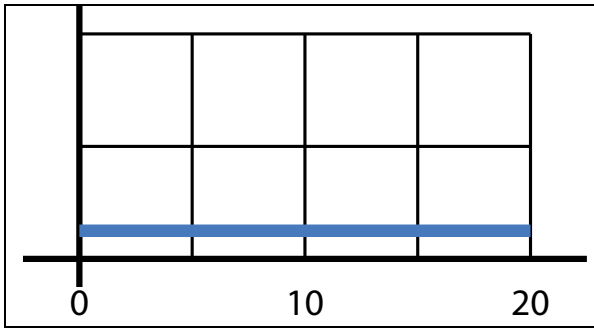
1pt max at zero
1 pt for straight line
1 pt for decreasing

B) There is no ambient or specular lighting (their values are 0), but a large *diffuse* component, and *Phong* shading..



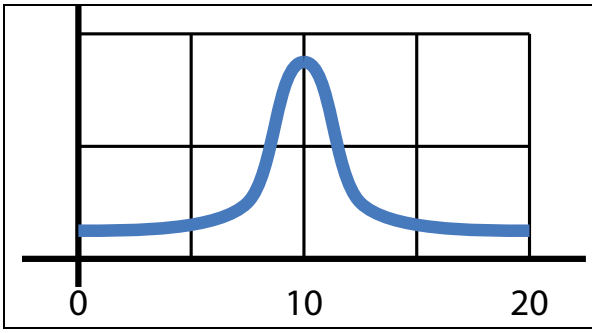
1pt max at zero
1pt for monotonic decrease
1 pt for curved
Note: having the wrong curve is OK in terms of points
(this picture was plotted using excel, since I was curious about the actual shape)
X=10,is 45degrees,.707 value

C) There is no ambient or diffuse lighting, but there is a large *specular* component with a very large exponent. Use *Gouraud* shading.



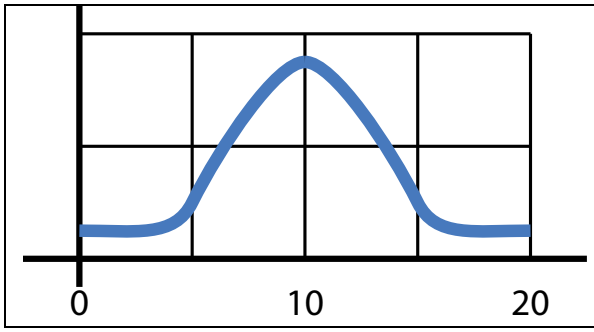
1 pt for beginning and end being the same
1 pt for straight line
1 pt for value being low

D) There is no ambient or diffuse lighting, but there is a large *specular* component with a very large exponent. Use *Phong* shading.



1 pt for begin and end being low and the same
1 pt for max in the middle
1 pt for a narrow hump

E) Exactly as part (D), but with a much smaller specular *exponent* (shininess value). What we are looking for here is the difference between this situation and (D).



1 pt for max in middle
2 pt for bump being wider than D